

**DORMAN Computer Program
(Study 2.5) Final Report
Volume II
User's Guide and Programmer's Guide**

Prepared by DATA PROCESSING SUBDIVISION
Information Processing Division
Engineering Science Operations

15 September 1973

(NASA-CR-137367) DORMAN COMPUTER PROGRAM
(STUDY 2.5). VOLUME 2: USER'S GUIDE
AND PROGRAMMER'S GUIDE Final Report
(Aerospace Corp., El Segundo, Calif.)

N74-19828

25 p HC \$7.75 95-p

CSCL 09B G3/08 15926

Unclas

95
Prepared for OFFICE OF MANNED SPACE FLIGHT
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Washington, D. C.

Contract No. NASW-2472



Systems Engineering Operations
THE AEROSPACE CORPORATION

DORMAN COMPUTER PROGRAM (STUDY 2. 5) FINAL REPORT

Volume II: User's Guide and Programmer's Guide

Prepared by
Data Processing Subdivision
Information Processing Division
Engineering Science Operations

15 September 1973

Systems Engineering Operations
THE AEROSPACE CORPORATION
El Segundo, California


Prepared for
OFFICE OF MANNED SPACE FLIGHT
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Washington, D. C.

Contract No. NASW-2472


DORMAN COMPUTER PROGRAM (STUDY 2.5) FINAL REPORT


Volume II: User's Guide and Programmer's Guide


Prepared by


Stanley T. Wray, Jr., Staff
Engineer
Advanced Application Staff
Data Processing Subdivision

Approved by


L. Sashkin, Director
Data Processing Subdivision
Information Processing Division


Robert R. Wolfe
NASA Study Director
Advanced Vehicle Systems
Directorate
Systems Planning Division


L. R. Sitney, Associate Group Director
Advanced Vehicle Systems Directorate
Systems Planning Division

FOREWORD

The DORMAN program was developed to create and modify a data bank containing data decks which serve as input to the DORCA Computer Program. Via a remote terminal a user can access the bank, extract any data deck, modify that deck, output the modified deck to be input to the DORCA program, and save the modified deck in the data bank.

This computer program is an assist in the utilization of the DORCA program. The program is dimensionless and operates almost entirely in integer mode. The program was developed on the CDC 6400/7600 complex at Aerospace Corporation, El Segundo, California for implementation on a UNIVAC 1108 computer.

The program has been delivered to the NASA Computing Facility located in Slidell, Louisiana at the direction of Vearl Huff (COR), NASA Headquarters, Washington, D. C. A data bank has also been delivered containing 24 DORCA data decks. Documentation provided under Study 2.5 DORCA Applications is:

- | | |
|------------|-------------------------------------|
| Volume I | Executive Summary |
| Volume II | User's Guide and Programmer's Guide |
| Volume III | Original DORMAN Data Bank |
| Volume IV | DORMAN Program Listing |

The format of this volume follows standard practice for User's and Programmer's Guides, which deviates from the format of technical reports. Sections, or subroutine definitions are not numbered to preclude confusing these types of numerics with the program code.

CONTENTS

| | |
|-----------------------------------|-----|
| ABSTRACT | iii |
| 1. USER'S GUIDE | 1 |
| General | 2 |
| Deck Definition | 2 |
| Tape Units | 2 |
| Beginning of Run | 4 |
| Errors | 5 |
| The Option - CREATE | 6 |
| The Option - USE | 8 |
| The Option - OPTION | 9 |
| The Option - SAVE | 11 |
| The Option - ADD | 13 |
| The Option - DELETE | 16 |
| The Option - EDIT | 17 |
| The Option - CONVERT | 19 |
| The Option - REPLACE | 21 |
| The Option - LIST | 23 |
| The Option - DONE | 26 |
| The Option - BATCH | 27 |
| More Complex Operations | 28 |
| 2. PROGRAMMER'S GUIDE | 31 |
| Main Program DORMAN | 32 |
| Subroutine ADD | 33 |
| Subroutine ADDER | 35 |
| Subroutine ADDX | 36 |
| Subroutine A1TA6 | 37 |
| Subroutine A6TA1 | 38 |
| Subroutine CKSYN | 39 |

PRECEDING PAGE BLANK NOT FILMED

CONTENTS (Continued)

| | |
|-----------------------------|----|
| Subroutine CONV | 40 |
| Subroutine COUNT | 41 |
| Subroutine DELCD | 42 |
| Subroutine DELET | 43 |
| Subroutine DELETE | 44 |
| Subroutine DIFDEC | 45 |
| Subroutine EDITDK | 46 |
| Subroutine EDITER | 47 |
| Subroutine EXTRAN | 48 |
| Subroutine FILBUF | 49 |
| Subroutine FNDBUF | 50 |
| Subroutine INCD | 51 |
| Subroutine INCRT | 53 |
| Subroutine INSERT | 54 |
| Subroutine INTBUF | 55 |
| Subroutine GETGEN | 56 |
| Subroutine LABLER | 57 |
| Subroutine LCRD | 58 |
| Subroutine LISTER | 59 |
| Subroutine LISTG | 60 |
| Subroutine LISTTC | 61 |
| Subroutine OPT | 62 |
| Subroutine OUTCD | 63 |
| Subroutine PACCON | 65 |
| Subroutine REPL | 66 |
| Subroutine REPLAC | 67 |
| Subroutine RESTOR | 69 |
| Subroutine SAVER | 70 |
| Subroutine SYNCBF | 71 |

CONTENTS (Continued)

| | |
|--|----|
| Subroutine SYNCDS | 73 |
| Subroutine SYNC1 | 74 |
| Subroutine TERM | 75 |
| Subroutine UNPAC | 76 |
| Subroutine UREAD | 77 |
| Subroutine USE | 78 |
| Subroutine USER | 79 |
| Subroutine VALUE | 80 |
| Format of Basic Data File | 81 |
| File Definitions | 83 |
| List of Commons with Variables | 84 |

FIGURE

| | | |
|----|------------------------------------|----|
| 1. | Segmentation/Linkage Map | 88 |
|----|------------------------------------|----|

TABLES

| | | |
|----|--------------------------------------|----|
| 1. | DORMAN Commands | 1 |
| 2. | Tapes Used by Commands | 3 |
| 3. | DORMAN Options/Subroutines | 32 |

1. USER'S GUIDE

DORMAN is a user oriented interactive program designed to run on the UNIVAC 1108 computer. Its vocabulary consists of eleven simple commands described in detail in this section. Each command has its own series of prompting requests for more information so that the user invokes one command at a time leading him down a trail of input information to complete a process via DORMAN.

The program manipulates a data file containing DORCA data decks. By coupling together a sequence of commands, the user can build a data file, add or delete decks from the data file, set up DORCA runs, modify existing DORCA data decks to study parametric variations in mission models, etc., all by sitting at a remote console terminal in an office miles away from the computer.

This section describes the commands available in DORMAN, the additional information required of the user, and provides some sample commands and sequences to do some more complex items. Table 1 is a summarization of the commands available.

Table 1. DORMAN Commands

| Command | Description |
|---------|--|
| CREATE | Create a new data base |
| USE | Use existing data base |
| OPTION | List options available |
| SAVE | Save a card file |
| ADD | Add a deck to the data base |
| DELETE | Delete a deck from the data base |
| EDIT | Build a mod deck via input |
| CONVERT | Difference two decks to get a mod deck |
| REPLACE | Replace a deck in data base |
| LIST | Request listings |
| DONE | Terminate run |
| BATCH | Print input command in output listing |

GENERAL

Before the various options in DORMAN are described, the following paragraphs will provide the user with some understanding of: the tapes used by DORMAN; the definition of various decks used by DORMAN; and how the program communicates with the user.

DECK DEFINITION

DORCA deck - a data deck of 80 column cards varying typically between 1200 and 2200 cards in length. This deck provides a basic constraint on the program in that the cards are divided into 8 fields of 10 columns each with no usage of the characters "=" and ";" in the deck.

Basic deck - a basic deck is a DORCA deck with an additional card on the front of the deck and a card on the back of the deck. The additional front card is of the form \$DECK NAME where NAME is a 12 letter identifier. The additional last card is of the form \$END OF DECK.

Mod deck - a deck that contains commands INSERT, DELETE and CHANGE with data cards that will convert a basic deck into another basic deck. As mod decks are usually smaller than the basic deck, DORMAN should use mod decks wherever possible.

Data base - the data base is a super sized collection of basic decks and mod decks arranged in a coherent form with a table of contents. The first card of a data base (or data file) is the \$VERSION card and the last card of the data base is \$END OF FILE.

TAPE UNITS

Table 2 shows the inter-relationship of tape units and the options available in DORMAN. The option CREATE accesses the ADD option and therefore will require the tapes used in the ADD option. Similarly the USE option is referred to by the options, SAVE, CONVERT, EDIT, and LIST. A tape designated as I is input only; as O is output only; as I/O is tape that can be both read and written; and as S is a scratch file used only by that option.

Table 2. Tapes Used by Commands

| Tape No. | Command | | | | | | | | | | |
|----------|---------|-----|--------|------|-----|--------|---------|---------|------|------|------|
| | CREATE | USE | OPTION | SAVE | ADD | DELETE | CONVERT | REPLACE | DONE | EDIT | LIST |
| 1 | | I | | I | I | I | I | I | | I | I |
| 2 | O | I | | I | I/O | I/O | | I/O | | | I |
| 3 | | I | | I | I/O | I/O | | I/O | | | I |
| 4 | | | | O | | | | | | | I |
| 5 | I | I | I | I | I | I | I | I | | I | I |
| 6 | O | O | O | O | O | O | O | O | O | O | O |
| 11 | | | | | I | | | | | | I |
| 12 | | | | O | | | | | | | I |
| 13 | | I | | I | I | | I | I | | I | I |
| 14 | | I/O | | I | I | | I | I | | I | I |
| 20 | | | | | | | | | | | O |
| 21 | I | | | | I | | | | | | I |
| 22 | | S | | | S | | S | | | S | I |
| 23 | | S | | | | | S | | | S | I |
| 24 | | | | O | | | | | | | I |
| 25 | | S | | | | | | | | | I |
| 26 | | S | | | | | | | | | I |
| 27 | | S | | | | | | | | | I |

Some tapes are declared external to DORMAN so that a user can communicate data to and from the program. The specific external tapes are: 1, 4, 11, 12, 20, 21, 24. The other tapes can be accessed by very clever individuals who have studied the manual and program listings very thoroughly. However, such practices are considered dangerous and the responsibility of the user.

The operations of DORMAN can be considered as invoking an option which reads data from specific tape units and produces results on other tape units. Thus a general understanding of the tape units and the options is recommended before the user attempts to execute the program. The program is protected to the extent that the original data base cannot be destroyed via the DORMAN program. To begin the processing the program demands the user specify a choice of two options, "CREATE" or "USE", initially to inform the program of where the data base is to be found-on tape 1 or tape 3.

The processing is highly automated, requesting the user to supply the minimum amount of information required for each step in the process.

BEGINNING OF RUN

When the user performs the necessary actions via the terminal to have the system load and execute the DORMAN program, the program signifies it is loaded and ready to begin processing by displaying

```
- - BEGIN DORMAN - -
```

```
DO YOU WISH TO CREATE A DATA FILE OR TO USE SUCH  
A FILE
```

```
ENTER CREATE OR USE
```

followed by the line:

```
ENTER OPTION REQUEST -
```

At this point the terminal is waiting for the word CREATE or USE to be entered right after the dash. Thereafter, the program will always return

to the line for option requests to permit entering any of the eleven commands available. Any input not understood by the program will produce a message

COMMAND NOT UNDERSTOOD - PLEASE RETRY

and cause the line

ENTER OPTION REQUEST -

to be displayed again.

If at any time the line

READY -

is displayed, the program is waiting for input data as described by a preceding line. At this point, the program will do nothing until the desired information has been transmitted.

ERRORS

During the execution of the various procedures in DORMAN, certain conditions arise leading to error messages at the terminal. Most occurrences are non-fatal and the program continues to operate as if the error had not occurred. The error messages are considered self-explanatory. When a fatal message is printed, the program terminates with a statement of fatal error. Then the general problem is one of damaged or nonexistent files or inadvertent system errors. The basic data file can be validated with the area of the document entitled Format of the Basic Data File.

THE OPTION - CREATE

When the word "CREATE" is entered in response to the first request, tape 2 is assigned for the creation of a data base. Any data base that may have been available on tape 1 will be ignored. A mini data base will be established on tape 2 consisting of the basic deck TEST and the mod deck TESTOR. Both decks are empty, but are required for DORMAN to function.

Program control is then passed into a portion of the "add decks to data base" option to add the first basic deck on tape 21. If the first card on tape 21 is not a \$DECK, said card will be generated via a query from DORMAN

PLEASE ENTER NAME FOR DECK

READY -

at which time the user is to supply a 12 letter name to be given to this deck. DORMAN will then state:

DECK FOUND - NAME

ENTER OK OR SKIP

READY -

The user can enter "OK" at which time DORMAN will copy the deck to a scratch file making sure that the \$END OF DECK card is present. If no such card is found before the end-of-file is encountered, the program will then state:

EOF FOUND - IF ACCEPTED, ENTER GO, OTHERWISE DONE

READY -

If the user elects to provide "DONE", the add process is abandoned and the user is left with the mini data base (control is passed back to the option request level). If the user gives a "GO", the \$END OF DECK card is generated and the message

NNNN CARDS FOUND IN DECK

providing the user with a count of the deck. The deck (correctly structured for the data base) is added to the data via a process that begins with the message

ADD DECK NAMED NAME TO DATA FILE

and terminates with the option request.

If the user had given a "SKIP" the deck would have been by-passed. There is no card count, but the \$END OF DECK check is performed. The logic then attempts to check for \$DECK card beginning the deck, whereupon the cycle repeats until a deck is added to the data base, or the word "DONE" is accepted on an end-of-file. Thus, the user has the capability of picking decks from a group of decks.

The CREATE option may be used only once at the beginning of a run. Sample sequence for CREATE (inputs underlined):

ENTER OPTION REQUEST - CREATE

DECK FOUND - CASE WILD 2

ENTER OK OR SKIP

READY - OK

1282 CARDS FOUND IN DECK

ADD DECK NAMED CASE WILD 2 TO DATA FILE

ENTER OPTION REQUEST -

THE OPTION - USE

When the word "USE" is entered in response to the first request, the message

CREATE OPTION IS BLOCKED

appears. This message never appears again. This is followed by:

IF YOU WISH TO HAVE A DECK EXTRACTED FROM THE BANK
ENTER THE NAME OF THE DECK, OTHERWISE DONE
READY -

The user may terminate the option by entering the word "DONE". Otherwise the name of a deck in the data bank is to be provided, and the program places the basic deck on tape 14.

If there is a mod deck existing on tape 13, and known to the program as the result of the EDIT or CONVERT options, the name of the deck on tape 13 would cause the basic deck corresponding to the mod deck to be placed on tape 14.

The USE option can be used as many times as desired. Sample sequence for USE (inputs underlined):

ENTER OPTION REQUEST - USE

CREATE OPTION IS BLOCKED

IF YOU WISH TO HAVE A DECK EXTRACTED FROM THE BANK
ENTER THE NAME OF THE DECK, OTHERWISE DONE

READY - CASE WILD 1

ENTER OPTION REQUEST -

THE OPTION - OPTION

When the word "OPTION" is entered in response to the option request, the following table is printed on the terminal:

OPTION LIST (SHORT FORM)

- | | | | |
|----------------|---------------|---------------|----------------|
| 1. CREATE | 2. USE (U) | 3. OPTION (O) | 4. SAVE (S) |
| 5. ADD (A) | 6. DELETE (D) | 7. EDIT (E) | 8. CONVERT (C) |
| 9. REPLACE (R) | 10. LIST (L) | 11. DONE | 12. TAPE LIST |

This is a list of the available options in the DORMAN program. The user may input the full word or the short form; e.g., SAVE, or S (the short form in parenthesis). The printout continues:

ENTER NUMBER OF OPTION TO BE EXPLAINED OR DONE

READY -

This option terminates if the word "DONE" is input. Otherwise the entry is checked for numeric in the range 1-12. If not acceptable, the user is informed

ILLEGAL ENTRY - TRY AGAIN

READY -

to supply 1-12 or "DONE".

The user will then be presented with a detailed description of each option 1-12 as per the individual number supplied. The program loops back for the next number or "DONE". Therefore, the only way to terminate this feature is to supply the word "DONE". The tape list is included so that the user is not forced to refer to the manual for a description of the tapes in the program.

Sample sequence for OPTION (inputs underlined):

ENTER OPTION REQUEST - OPTION

OPTION LIST (SHORT FORM)

- | | | | |
|----------------|---------------|---------------|----------------|
| 1. CREATE | 2. USE (U) | 3. OPTION (O) | 4. SAVE (S) |
| 5. ADD (A) | 6. DELETE (D) | 7. EDIT (E) | 8. CONVERT (C) |
| 9. REPLACE (R) | 10. LIST (L) | 11. DONE | 12. TAPE LIST |

ENTER NUMBER OF OPTION TO BE EXPLAINED OR DONE

READY - DONE

ENTER OPTION REQUEST -

THE OPTION - SAVE

When the word "SAVE" is entered in response to the option request, DORMAN asks for a definition of what is to be saved by:

ENTER SAVE OPTION (DATA BASE, DORCA OR MOD DECK)

READY -

If the response is "DATA BASE", DORMAN will save the current data base on tape 4, unless no additions or deletions have been made to the data base as yet. The user will be requested to supply an identifier for the new data base by

ENTER VERSION IDENTIFIER

READY -

whereupon the user would supply a six character identifier.

If the response is "DORCA", the program will ask for the name of the deck to be input to DORCA by:

ENTER NAME OF DECK TO BE USED FOR DORCA INPUT

READY -

After the name is supplied, the program checks to see if a basic deck already exists on tape 14. If there is no deck, the desired deck will be placed on tape 14 as a basic deck and then saved on tape 12. If there is a deck and the name agrees with the entry, the basic deck is saved on tape 12. If the names do not agree, the program protects the user by requesting permission to destroy the deck on tape 14 by:

DECK IS NOT BASIC

DESTRUCT PERMISSION REQUIRED

ENTER YES OR NO

READY -

If the response is "NO" the SAVE option is terminated. If the response is "YES", the desired deck will be placed on tape 14 as a basic deck and then saved on tape 12.

If the save option is "MOD DECK", the program asks for the name of the mod deck to be saved:

ENTER NAME OF DECK TO BE SAVED AS MOD DECK

READY -

The name entered is checked against the name of the mod deck currently on tape 13. If there is no deck or the names do not agree, the program responds with

DECK NAMES DO NOT MATCH

MOD DECK NOT AVAILABLE

REQUEST DENIED

and the option is terminated.

If the names match, the mod deck is saved on tape 24.

The SAVE option can be used as many times as desired to stack mod decks on tape 24.

Sample sequence for SAVE (inputs underlined):

ENTER OPTION REQUEST - SAVE

ENTER SAVE OPTION (DATA BASE, DORCA OR MOD DECK)

READY - DORCA

ENTER NAME OF DECK TO BE USED FOR DORCA INPUT

READY - CASE WILD 2

ENTER OPTION REQUEST -

THE OPTION - ADD

When the word "ADD" is entered in response to the option request, the program wishes to know which tape it will find the deck to be added to the data base. The user is requested by:

ENTER INPUT

OR BASIC

OR MOD DECK

READY -

Where "INPUT" implies one or more decks on tape 21 (the primary external source of decks), "BASIC" implies a generated basic deck residing on tape 14, and "MOD DECK" implies a mod deck on tape 13 (generated by DORMAN by deck differencing or generated by the user in the EDIT option) or input on tape 11. DORMAN will clarify the mod deck by printing:

IS MOD DECK GENERATED OR INPUT ON TAPE 11

READY -

Tape 11 is used if the user supplies "INPUT", otherwise tape 13 is used.

Program control is then passed into a portion of the "add decks to data base" option to add the first basic deck on tape 21; e.g., if the first card on tape 21 is not a \$DECK, said card will be generated via a query from DORMAN

PLEASE ENTER NAME FOR DECK

READY -

at which time the user is to supply a 12 letter name to be given to this deck. DORMAN will then state:

DECK FOUND - NAME

ENTER OK OR SKIP

READY -

The user can enter "OK" at which time DORMAN will copy the deck to a scratch file making sure that the \$END OF DECK card is present. If no such card is found before the end-of-file is encountered, the program will then state:

EOF FOUND - IF ACCEPTED, ENTER GO, OTHERWISE DONE
READY -

If the user elects to provide "DONE", the add process is abandoned and the user is left with the original data base (control is passed back to the option request level). If the user gives a "GO", the \$END OF DECK card is generated and the message

NNNN CARDS FOUND IN DECK

providing the user with a count of the deck.

The deck (correctly structured for the data base) is added to the data via a process that begins with the message

ADD DECK NAMED NAME TO DATA FILE

and terminates with the option request.

If the user had given a "SKIP", the deck would have been by-passed. There is no card count, but the \$END OF DECK check is performed. The logic then attempts to check for \$DECK card beginning the deck, whereupon the cycle repeats until a deck is added to the data base, or the word "DONE" is accepted on an end-of-file.

The ADD option adds one deck to the data base each time it is invoked. The process is completed when the user is requested to enter another option. The ADD option can be used as many times as necessary.

Sample sequence for ADD (inputs underlined):

ENTER OPTION REQUEST - ADD

ENTER INPUT

OR BASIC

OR MOD DECK

READY - INPUT

DECK FOUND - CASE WILD 2

ENTER OK OR SKIP

READY - OK

1282 CARDS FOUND IN DECK

ADD DECK NAMED CASE WILD 2 TO DATA FILE

ENTER OPTION REQUEST -

THE OPTION - DELETE

When the word "DELETE" is entered in response to the option request, the program asks for the name of the deck to be deleted from the data base by:

WHAT DECK IS TO BE DELETED

READY -

The user supplies the name of a deck in the data bank and the program will delete the deck stating:

DELETE DECK NAMED NAME FROM DATA FILE

After the deck is deleted, the program will give the card count of the deleted deck

NNNN CARDS DELETED

and return to the option request level.

Sample sequence for DELETE (inputs underlined):

ENTER OPTION REQUEST - DELETE

WHAT DECK IS TO BE DELETED

READY - CASE WILD 1

DELETE DECK NAMED CASE WILD 1 FROM DATA FILE

1289 CARDS DELETED

ENTER OPTION REQUEST -

THE OPTION - EDIT

When the word "EDIT" is entered in response to the option request, DORMAN responds with

BEGIN EDIT OPERATION

ENTER NAME OF NEW DECK

READY -

where the user will enter the name of the new deck to be built. The program then requests the name of the reference deck:

ENTER NAME OF REFERENCE DECK

READY -

The name provided for the reference deck is checked against the name of the basic deck currently on tape 14, if any. If there is no basic deck on tape 14, the deck will be extracted from the data base and put on tape 14. If the names match for the deck on tape 14, the program proceeds immediately to the next step. If the names do not match, the program protects the user by:

REFERENCE IS NOT BASIC

DESTRUCT PERMISSION REQUIRED

ENTER YES OR NO

READY -

If the user enters "NO", the EDIT option is terminated. If the user enters "YES", tape 14 will have the correct reference deck put on the tape.

The EDIT option then begins with:

ENTER MOD CARDS

ENTER DONE WHEN FINISHED

READY -

The user now inputs cards with each "READY -" as mod cards \$ADD = N, \$DELETE = N1 = N2, \$CHANGE = N1 = X = Y, and DORCA data cards. The processing is card by card. The mod cards cause the editing operation to position on or after the card N or N1 or N2 depending upon the modification to be made. Any non-mod card is a DORCA data card and is an automatic insert at the current position in the card file. The \$CHANGE specifies the editor to find card N1, and within the card find the 10 column entry X and change it to the 10 column entry Y; only one item will be altered, the first on the card. Either X or Y could be blank. The list option LIST = N1 = N2 is available to look at the reference deck, cards N1 through N2. Operations may be done in any order except data cards must follow their positioning card. When the word "DONE" is entered, the user is put back at the option request level with a mod deck on tape 13. (Only one deck is on Tape 13)

Sample sequence for EDIT (inputs underlined):

ENTER OPTION REQUEST - EDIT

ENTER NAME OF NEW DECK

READY - WILDX

ENTER NAME OF REFERENCE DECK

READY - CASE WILD 1

ENTER MOD CARDS

ENTER DONE WHEN FINISHED

READY - \$INSERT = 2

READY - = TEST

READY - LIST = 1 = 1

1 COMMENT

READY - DONE

ENTER OPTION REQUEST -

THE OPTION - CONVERT

This option will take two basic decks and go through a differencing procedure to create a mod deck that represents the modifications to be made to one of the basic decks to obtain the other basic deck. The user can then replace the large basic deck with the smaller mod deck in the data base. The reference deck is the basic deck that will remain in the data base. The final deck is that deck that could be replaced in the data base by the smaller mod deck.

When the word "CONVERT" is entered in response to the option request, the program requests the name of the deck to be used as the reference deck by printing:

ENTER NAME OF REFERENCE DECK

READY -

After being supplied the name for the reference deck, the program requests the name of the final deck by:

ENTER NAME OF FINAL DECK

READY -

DORMAN will then extract the two decks from the bank by applying the USE feature automatically or if both decks are basic decks in the bank, they will be copied. The reference deck is on tape 22, and the final deck is on tape 23. The deck differencing capability of DORMAN will be used to create a mod deck that will represent the required action to convert the reference deck to the final deck by the application of a mod deck. This mod deck could replace the final deck in the data bank. This technique permits the user to compact the data bank via an automated generation of a mod deck.

The user is informed when the deck differencing process is about to begin by the appearance of the first card of the new mod deck

\$DECK NAME2 USES NAME1 TIME DATA

where NAME1 is the name of the reference deck, NAME2 is the name of the final deck and TIME and DATE are time of day and date of day.

The user is then requested to supply sync cards (up to 20 pairs) which is terminated with the word "DONE". The process can be lengthy and the user is informed of completion when the option request appears.

If the program prints

THERE WILL BE A WAIT FOR OUT OF CORE SEARCH FOR MATCH
the user might be required to use the sync cards to force a match. The numbered pair tells the program to establish a boundary in each deck to achieve a match (if there is none) with card N1 in the reference deck and card N2 in the final deck. The pairs of entries for sync cards are in sequence. The use of sync cards can reduce the size of a mod deck and reduce the computer time for out of core match.

The CONVERT option can be used as many times as necessary.

Sample sequence for CONVERT (inputs underlined):

ENTER OPTION REQUEST - CONVERT

ENTER NAME OR REFERENCE DECK

READY - CASE WILD 1

ENTER NAME OF FINAL DECK

READY - CASE WILD 2

\$DECK CASE WILD 1 USES CASE WILD 2 15:06:00 06/01/73

ENTER SYNC CARDS

READY - DONE

ENTER OPTION REQUEST -

THE OPTION - REPLACE

When the word "REPLACE" is entered in response to the option request, DORMAN requests the name of the deck to be put in the data file:

ENTER NAME OF DECK TO BE USED

READY -

The name entered is identified as a mod deck on tape 13 or a basic deck on tape 14. If the deck is not present, the request terminates with:

ORIGINAL DECK NOT FOUND

The program makes a check by requesting the name of the deck in the data file by:

ENTER NAME OF DECK TO BE REPLACED

READY -

The two names are compared to prevent errors. If the names do not match, the option terminates with the message:

DECK NAMES DO NOT MATCH

REQUEST REJECTED - USE DELETE AND ADD

Now DORMAN is ready to replace a mod or basic deck in the data file with a mod or basic deck generated by the user at the terminal.

REPLACE DECK NAMED NAME ON DATA FILE

The process terminates when the user is returned to the option request level. The option can be used as many times as desired.

Sample sequence for REPLACE (inputs underlined):

ENTER OPTION REQUEST - REPLACE

ENTER NAME OF DECK TO BE USED

READY - CASE WILD 2

ENTER NAME OF DECK TO BE REPLACED

READY - CASE WILD 2

REPLACE DECK NAMED CASE WILD 2 ON DATA FILE

ENTER OPTION REQUEST -

THE OPTION - LIST

When the word "LIST" is entered in response to the option request, DORMAN asks the user to supply which list option he desires:

ENTER LIST OPTION (CONTENTS, GENEALOGY, COUNT CARDS,
OR PRINT)

READY -

If the entry does not correspond to any item within the parenthesis, the option is terminated and the user is put back on the option request level.

If the option is "CONTENTS", then the table of contents of the current data base is printed on the terminal.

If the option is "GENEALOGY", the program will give the message

ENTER NAME OF DECK DESIRED

READY -

expecting the user to supply the name of a data deck in the data base.

DORMAN will then print the chain of mod decks terminating with basic deck (this is referred to as the genealogy of a deck).

The other three list options require definition of a tape to be read as a source of cards or card decks. DORMAN prints the message

ENTER NAME OF DECK

OR FILE NUMBER

OR CURRENT FILE NAME

OR DONE

READY -

waiting for the name of a deck to be extracted from the data file, a tape number 1-4, 11-14, 21-27, the word "BASIC" for the current data base, the word "FINAL" for the current basic deck, or the word "DONE" to terminate the option.

The option "COUNT" against tape 14 will produce the output:

TAPE 14 WITH \$DECK NAME CONTAINS NNNN CARDS

The option "CARDS" against tape 14 will require the user to supply the number of the first card to be printed from the deck on tape 14 in response to

ENTER NUMBER OF FIRST CARD

READY -

and the number of the last card via:

ENTER NUMBER OF LAST CARD

READY -

All cards beginning with the first and through the last will be listed. This is useful for snapshots of decks. The first card of a deck is card 0 (\$DECK).

The option "PRINT" against tape 14, will produce a batch printout on tape 20 for the entire deck on tape 14 with card numbers on the left margin.

Sample sequence for LIST (inputs underlined):

ENTER OPTION REQUEST - LIST

ENTER LIST OPTION (CONTENTS, GENEALOGY, COUNT, CARDS, PRINT)

READY - COUNT

ENTER NAME OF DECK

OR FILE NUMBER

OR CURRENT FILE NAME

OR DONE

READY - 14

TAPE 14 WITH \$DECK CASE WILD1 CONTAINS 1289 CARDS

ENTER NAME OF DECK

OR FILE NUMBER

OR CURRENT FILE NAME

OR DONE

READY - DONE

ENTER OPTION REQUEST -

THE OPTION - DONE

When the option request is "DONE", the DORMAN program terminates processing with the printout:

-- END DORMAN -- RUN TERMINATED --

All knowledge of the information existing on files is lost. The program cannot be reentered to "save" data. Some information is retrievable, if the user is familiar with the area of this document listed in the table of contents as File Definitions.

Sample sequence for DONE (inputs underlined):

ENTER OPTION REQUEST - DONE

-- END DORMAN -- RUN TERMINATED --

THE OPTION - BATCH

The purpose of the command "BATCH" is to provide traceability for runs submitted in a batch mode, or to permit the user to keep track of the program's functions, if the user decides to push the terminal operation by inputting commands at a higher rate than the computer system is capable of responding to. After the command "BATCH" is entered, the program will print all input commands immediately after the corresponding request for input. Thus, the printout on the terminal or a batch run will show the exact inputs provided to DORMAN as they are being processed.

MORE COMPLEX OPERATIONS

The user can link together several option requests to achieve an overall aim.

Samples are:

1. Extract a deck for a DORCA run.
USE (provide name of deck)
SAVE (specify DORCA)
2. CREATE a data base.
CREATE
ADD
ADD
ADD
.
.
.
N decks
3. Convert a deck, compare the basic and the mod deck and decide to replace the basic in the data file.
CONVERT
LIST (count on tape 13-mod deck)
LIST (count on tape 14-basic deck)
REPLACE
4. Modify a deck for a DORCA run and save the mod deck.
USE
EDIT

SAVE or ADD (mod deck)

SAVE (data base)

SAVE (DORCA)

5. Use an external mod deck for DORCA run.

ADD (mod deck on tape 11)

USE

SAVE (DORCA)

2. PROGRAMMER'S GUIDE

This section is provided as documentation of the program code for reference in the future, should the need arise to modify the code or the basic procedures embedded in DORMAN. The following pages give detailed descriptions of each subroutine in the program. There are several pages devoted to the description of the basic data file used in DORMAN. There is a description of how the files (19 in all) are used in DORMAN, and there is a list of the labeled common blocks with definition of variables. The section terminates with a segment linkage map (Figure 1) of the major program structure, excluding frequently used routines: INCD, OUTCD, INTBUF, etc.

The routine ERTRAN is a UNIVAC 1108 system routine providing time of day (A6 format - HHMMSS) and date (A6 format - MMDDYY)

The programmers responsible for coding the program were B. J. Gold, G. W. Timpson, V. V. Voit and S. T. Wray, Jr.

PRECEDING PAGE BLANK NOT FILMED

MAIN PROGRAM DORMAN

The main program of DORMAN initializes the program, responds to input commands via a terminal to call subroutines to perform the specified activity, recovers from errors if possible, and terminates the run upon input command. Table 3 describes the options with their associated subroutines. The short form can be used instead of the command.

Table 3. DORMAN Options/Subroutines

| Input Command | Short Form | Description | Subroutine Called |
|---------------|------------|--|-------------------|
| CREATE | - | Create a new data base | INCRT |
| USE | U | Use existing data base | USER |
| OPTION | O | List options available | OPT |
| SAVE | S | Save a card file | SAVER |
| ADD | A | Add a deck to data base | ADDER |
| DELETE | D | Delete a deck from data base | DELET |
| EDIT | E | Build a mod deck via input | EDITER |
| CONVERT | C | Difference two decks to get a mod deck | CONV |
| REPLACE | R | Replace a deck in data base | REPL |
| LIST | L | Request listings | LISTER |
| DONE | - | Terminate run/option | TERM |
| BATCH | - | Input commands are printed | - |

The program executes in a looping fashion; after each request is processed, the user is queried for the next request.

SUBROUTINE ADD

USAGE

CALL ADD (IN1, IN2, OUT)

where

IN1 is the file on which all basic data resides,

IN2 is the file containing the deck to be added,

OUT is the file on which the expanded data will be written.

Files IN1 and OUT are rewound by this routine upon entry. File IN2 is not rewound but is assumed to be correctly positioned at the beginning of the deck to be added.

The first card of the added deck must be in the standard format:

\$DECK NAME1

or

\$DECK NAME1 USES NAME2

The subroutine checks to see that NAME1 does not already exist on the basic data file IN1 but that NAME2 (if present) does. The added deck must end with the standard format card:

\$END OF DECK.

The added deck, whether a mod or basic deck, is always inserted into the data file after the last existing mod deck and before the first existing basic deck. An additional entry, consisting of a new \$DECK card plus the date and time in Hollerith, is inserted into the table of contents heading the output file OUT. This entry is at the very beginning of the table if the new deck is a mod deck, at the very end if a basic deck.

Error messages are printed on file FERR. If an error occurred, ADD sets ERFLAG = 1 upon return; otherwise, ERFLAG = 0. Possible errors are:

1. First card of added deck is not \$DECK.
2. Deck to be added already exists on data file.

3. Deck to be added requires or uses a deck which is not listed in the table of contents.

This subroutine always reads file IN2 until the \$END OF DECK card is found (until a read error prematurely terminates the routine). If an error occurs, the routine advances file IN2 to the end of the current deck, as indicated by the \$END OF DECK card.

SUBROUTINE ADDER

This routine controls the operation to add decks to an existing data base file.

USAGE

CALL ADDER

This routine communicates with the terminal to obtain the information of which file has the deck to be added. The input options (file) are: "INPUT" (21) or "BASIC" (14) or "MOD DECK" (11 or 13). If the response is "MOD DECK", the routine then asks for clarification where the word "INPUT" is tape 11. Tapes 11, 13 and 14 are always rewound; tape 21 is rewound on first entry only. The subroutine ADDX is used to further clarify the situation before ADDX uses the subroutine ADD to actually do the operation.

SUBROUTINE ADDX

This routine controls the process of adding one deck at a time to a data base. It provides quality assurance that the decks are in DORMAN format.

USAGE

CALL ADDX (INDKF)

where

INDKF is the tape number for the input file, usually 11, 13, 14 or 21.

Due to tape positioning problems elsewhere in DORMAN, the deck on INDKF is copied to tape 22. If the first card is not a \$DECK card, the user is requested to supply the name for the deck. Then the user is informed that deck so-and-so has been found, enter "OK" or "SKIP". If "OK" is entered, the deck is accepted and copied to tape 22. If an end-of-file is detected before a \$END OF DECK card, the user is given the option of terminating the process or finally accepting the deck. The routine will then add a \$END OF DECK card and call the routine ADD to put the deck in the data base. If the entry had been "SKIP", the deck would have been passed over until the next \$DECK card was encountered. Then the process resumes with enter "OK" or "SKIP". This routine adds only one deck. Tape 22 and the data base are the only files rewound.

SUBROUTINE A1TA6

This routine converts Hollerith strings from 84A1 format to 14A6 format using ENCODE .

USAGE

CALL A1TA6 (CA84, CA14)

where

CA84 is an 84 word array of A1 format.

CA14 is a 14 word array of A6 format.

This routine is compatible with the systems on the UNIVAC 1108, CDC 6400 and CDC 7600 computers.

SUBROUTINE A6TA1

This routine converts Hollerith strings from 14A6 format to 84A1 format using DECODE.

USAGE

CALL A6TA1 (CA14, CA84)

where

CA14 is a 14 word array of A6 format.

CA84 is an 84 word array of A1 format.

This routine is compatible with the systems on the UNIVAC 1108, CDC 6400 and CDC 7600 computers.

SUBROUTINE CKSYN

This routine obtains the current pair of sync card indices.

USAGE

CALL CKSYN (ICD1, ICD2, N1, N2)

where

ICD1 is the sync index to file 1 in DIFDEC.

ICD2 is the sync index to file 2 in DIFDEC.

N1 is the index to the current sync pair in the
 ISYN array.

N2 is a flag to advance the pointer N1 by 1 if N2 is
 not zero.

The values for ICD1 and ICD2 are obtained from the array ISYN
passed via common BFRS.

SUBROUTINE CONV

This routine controls the conversion operation that converts a basic deck into a mod deck by differencing the basic deck with a reference deck.

USAGE

CALL CONV

This routine communicates with a terminal to obtain the name of the original basic deck and the name of the reference basic deck. Both decks must exist in the data base file either as mod decks or as basic decks. The routine USE is called to put the basic decks on tape 22 and tape 23. The basic deck for the reference deck is on tape 22, and the basic deck for the original deck is on tape 23. The routine DIFDEC is used to generate the mod deck on tape 13, with sync cards being supplied from the user at the terminal.

SUBROUTINE COUNT

This routine counts cards on a specified file.

USAGE

CALL COUNT (NFILE)

where

NFILE is the tape number of the unit to be counted.

The unit NFILE is rewound and the key "NO NAME" is set up. The first card on NFILE is read; and, if the first card is a \$DECK card, the key is replaced with the deck name. The unit is read to count cards until an end-of-file is sensed, and then the key is printed with the card count. The routine assumes only one deck on the unit and therefore counts all cards on the unit.

SUBROUTINE DELCD

This routine is used to delete cards during a deck differencing operation.

USAGE

CALL DELCD (ISP)

where

ISP is the card number in file 1 just after the deletion process.

The other necessary variables are passed via common.

The delete card ~~\$\$\$~~DELETE N1 N2 is issued to the difference deck file FOUT and the routine RESTOR is used to advance the file FIN1, if the ISPth card is not in the buffer area BUF1.

SUBROUTINE DELET

This routine controls the deletion of a deck from the data base.

USAGE

CALL DELET

This routine DELET communicates with a terminal to determine the name of the deck to be deleted. The subroutine DELETE is called to do the delete operation. If an error was detected by DELETE, the desired deck was not deleted from the data base.

SUBROUTINE DELETE

This routine deletes the name deck from the indicated data file.

USAGE

CALL DELETE (NAME, IN1, OUT)

where

NAME is the name of the deck to be deleted (2 words,
 2A6 format).

IN1 is the file containing all data.

OUT is the file on which the revised data is written.

Files IN1 and OUT are rewound by DELETE before deletion and copying begin. Both the specified deck and its entry in the table of contents are deleted. The user must make sure that no deck remaining on the data file uses the deleted deck, or an error stop will occur.

Upon return from DELETE, ERFLAG = 0 if there was no error; ERFLAG = 1 if an error occurred. Error messages are written on file FERR. Possible errors are:

1. Deck to be deleted cannot be found on IN1.
2. Deck to be deleted is required by another deck on IN1.

SUBROUTINE DIFDEC

This routine controls the deck differencing process.

USAGE

CALL DIFDEC (FIN1, FIN2, FOUT, FSYN)

where

- FIN1 is the tape number of the file containing the master reference basic deck (NAME1).
- FIN2 is the tape number of the file containing the basic deck (NAME2) which could be generated by the application of a mod deck upon the master reference deck.
- FOUT is the tape number where the mod deck derived by the deck differencing process is to be output.
- FSYN is the tape number to be read for input of sync cards (set elsewhere in the program to be the console unit 5).

This routine calls SYNCDS to obtain a maximum of 20 sync points in the two decks, and then fills the two buffers BUF1 and BUF2. The first card of the mod deck is generated as

\$DECK NAME2 USE NAME1 TIME DATE

and placed on the file FOUT.

Two sets of pointers are maintained for both buffers: LIN1, LIN2 are the card numbers since the beginning of each file; CIN1, CIN2 are the current positions in the buffers for the same pair of cards.

Both buffers are advanced in card pairs, comparing one card in BUF1 against one card in BUF2. The buffers are refilled as necessary until the \$END OF DECK is reached in both files.

If no match is found for a pair of cards, a check is made for sync boundaries and either cards are inserted from FIN2, cards are deleted from FIN1; or a match is sought by calling SYNC1.

SUBROUTINE EDITDK

This routine performs the edit function of DORMAN.

USAGE

CALL EDITDK (FIN1, FIN2, FOUT)

where

FIN1 is the file containing a basic deck.

FIN2 is the input file containing a mod deck.

FOUT is the output file which will contain a basic deck generated by the action of the mod deck on the input basic deck.

The first card of a basic deck is \$DECK NAME. The first card of a mod deck is \$DECK NAME1 USES NAME2. The input basic deck is NAME2, and the output basic deck is NAME1. The last card of any deck is \$END OF DECK.

Any card not an EDIT command will be inserted at the current input basic deck position in the output basic deck. The EDIT commands are primarily file positioning commands for the input basic deck, and the modifications must be ordered in sequence. The EDIT commands are:

```
$INSERT     100
$DELETE     100                200
$CHANGE     100    =    3    =    SINGLE
```

Insertion is after the card named. Deletion is one or more cards. Change is on the card named, find the first entry and change to the second. Deletions may occur before insertions. Multiple changes are permitted and changes may occur before insertions. Therefore, a mod deck consists of many inserts, deletions and changes in an ordered process of converting one deck into another.

In the event an error is detected, an error message will be written on file, FERR; and the output file FOUT will be set negative as an error indication to the calling routine. The routine will attempt to yield all error messages.

SUBROUTINE EDITER

This routine controls the operation of building mod decks via terminal input to a format compatible with EDITDK.

USAGE

CALL EDITER

This routine communicates with a terminal to obtain the name of the mod deck. Then the user supplies the name of the basic deck that this mod deck will be applied against. This routine then checks to see if the basic deck is on tape 14. If not, a guard against an error requires the user to supply destruct permission to replace the basic deck on tape 14. The request for editing is terminated if the response is "NO". Then the initial mod deck is generated on tape 22 consisting of a \$DECK card and a \$END OF DECK card. As the deck is built up, it will be written onto tape 23 unless the mods are input out of order. In that case, the remainder of the deck is copied to tape 23 and the process begins anew with the mods being put on tape 22. The two units swap back and forth as necessary.

The user is told to input mods. The processing is card by card. The first card on tape 22 is copied to tape 23. If an input card is not a mod card of the form \$INSERT, \$CHANGE, \$DELETE, etc., the input card is placed on tape 23. If the card is a standard format, the numerics are extracted, and the file positioning is done so as to place this card in the sequential position in the mod deck. Mods to the same place in the basic deck are permitted, in accordance with the following rules:

1. Deletions are not permitted to overlap.
2. Changes are done before deletions or insertions.
3. Insertions are done in order of input.
4. Multiple changes are done in order of input.

Any card in the basic deck on tape 14 can be listed on the console by entering the phrase "LIST = 100" or "LIST = 100 = 103" to list one or more cards, beginning with card 100. The routine LCRD does the listing operation.

SUBROUTINE EXTRAN

This routine calls the UNIVAC 1108 System routine ERTRAN to obtain the current data and time from the computer.

USAGE

CALL EXTRAN (TIME, DATE)

where

TIME is a 2 word array (2A6 format) returned as
 HHMMSS

DATE is a 2 word array (2A6 format) returned as
 MMDDYY

SUBROUTINE FILBUF

This routine controls the filling of a card image buffer for DIFDEC.

USAGE

CALL FILBUF (BEGIN, END, MAX, BUF, FILE)

where

BEGIN is the location where the next card is to go in the
 buffer.

END is the location of the last card put in the buffer
 upon return.

MAX is capacity of the buffer in cards
 (END \leq MAX).

BUF is the buffer.

FILE is the tape to be read for more card images.

An internal request to add cards to a buffer that is already full will produce a fatal termination of DORMAN. The variable END will be equal to MAX, except when an end of file is encountered or a \$END OF FILE card. Then END will point to the \$END OF FILE card in either case.

SUBROUTINE FNDBUF

This routine is used to locate and protect files.

USAGE

CALL FNDBUF (IUNIT, IX, TY)

where

IUNIT is the tape number to be verified.

IX is the mode of UNIT (1, 2 or 3).

IY is the index of UNIT in the array IACT
(1, 2 or 3)

This routine searches the active table, IACT, to verify IUNIT is a currently active file. If the file is not active, the program is fatally terminated. Otherwise, the pointer and the mode are returned to the user.

SUBROUTINE INCD

This routine is used to do all read operations in DORMAN.

USAGE

CALL INCD (CARD, FILE)

where

CARD is always a 14A6 formatted array.

FILE is the tape number to be read for the card.

There are three different input formats handled by INCD. They are as follows:

- Mode 1 File is packed tightly with "=" as a 10 column tabbing symbol and ";" as a card terminating symbol. Input file is blocked in blocks of 4620 characters.
- Mode 2 File is standard FORTRAN input file.
- Mode 3 File is possibly packed with "=" as 10 column tabbing symbol.

The terminal input (TAPE 5) is designated as Mode 3 and not subject to any other constraints. The routine FNDBUF is called to get the mode of FILE and pointer to the array IACT. Then the routine branches to process Mode 1, 2 or 3.

Mode 1 processing:

After the end-of-file check, the buffer is filled with 4620 characters, the first 84 are extracted and a card image is expanded by UNPAC and converted to 14A6 format. The remaining characters are moved down in the working array for the next call. The end-of-file status is checked. The routine returns to the user. On subsequent entries, the next 84 characters are added to the remainder and the cycle repeats.

Mode 2 processing:

If the flag for end-of-file is set, the routine is terminated with a non-fatal error message. A call to UREAD produces a card which is checked for end-of-file. The routine returns to the user.

Mode 3 processing:

After checking for prior end-of-file, UREAD is called to obtain the next card which is unpacked by UNPAC and then checked for end-of-file. The routine returns to the user.

SUBROUTINE INCRT

This routine controls the operation of creating a data base file.

USAGE

CALL INCRT

This routine forms an abbreviated data base on tape 2 consisting of a basic deck named TEST and a mod deck named TESTOR. The data base is complete with data base label, table of contents, two card decks, and a \$END OF FILE terminator. The routine ADDX is called to add the first deck on tape 21 to the data base at the user's option. Once the data base contains one real deck, the user assumes control for adding all other decks on tape 21.

SUBROUTINE INSERT

This routine is used to add inserted cards to the file FOUT during a deck differencing operation.

USAGE

CALL INSERT (NSTP, I I)

where

NSTP is the number of the card in file 2 just after the
 insertion process.

I I is a flag with 2 states:
 = 0 implies an insert only.
 ≠ 0 implies a deletion with an insert.

The other necessary variables are passed via common.

If I I = 0 the card \$INSERT N1 is issued to the difference deck file FOUT. The cards to be inserted are copied from the buffer BUF2 to the file FOUT until the card NSTP is reached, whereupon the process is terminated.

The buffer BUF2 is filled by FILBUF and copied to the file as many times as necessary to position NSTPth card in the buffer.

SUBROUTINE INTBUF

This routine is used to control the buffer allocation in DORMAN by initializing 3 files as being active.

USAGE

CALL INTBUF (IU, IR)

where

IU is a 3 word array containing tape numbers or zero.

IR is a 3 word array containing the indicator:

0 read only request

1 write only request

2 read or write request

Each unit in the IU table is located in the IUTBL array and the entries in the IACT table are initialized as if the file were rewound. (This is critical only for Mode 1 files). The words in IACT (I, J) are:

J is 1, 2, or 3 for each of 3 files.

I = 1 Unit number

I = 2 Mode

I = 3 Read/write request

I = 4 End-of-file status, OK = 0

I = 5 Buffer Index, 1 or 2

I = 6 Work area index, 1 or 2

I = 7 Buffer card counter, 1 - 55

I = 8 Work area character counter.

SUBROUTINE GETGEN

GETGEN gets the genealogy of a specified data deck in the data base.

USAGE

CALL GETGEN (NAME, IFILE, LIST, NLIST)

where

NAME is the name of the data deck (2 words, 2A6 format).

IFILE is the FORTRAN logical number of the file on which
the data base resides.

LIST will contain upon return, the names (2 words/name)
of the decks in the genealogy, starting with NAME
itself. LIST is an array of dimension 2xN.

NLIST will contain upon return, the number of entries in
array LIST.

Tape IFILE is rewound upon entry to GETGEN.

If any of the decks in the genealogy are missing, GETGEN sets the
common variable ERFLAG = 1 and returns. If there is no error,
ERFLAG = 0. In the event of errors, error messages are printed on
the error file FERR.

SUBROUTINE LABLER

This routine is used to write final saved data base files on tape 4.

USAGE

CALL LABLER (IVER, FILE1, FILE2)

where

IVER is a 6 character version identifier.

FILE1 is tape number where the data base currently is.

FILE2 is the tape number where the data base is to be
saved (by convention, tape 4).

The new version card is written on FILE2 with new version ID, new date, and new clock time. The first card on FILE1 is skipped and the remainder of FILE1 is copied to FILE2.

SUBROUTINE LCRD

This routine prints cards with sequence numbers.

USAGE

CALL LCRD (FILE, ISTRT, ISTOP, PRTFIL)

where

FILE is the number of the tape containing the card deck.

ISTRT is the number of the first card to be listed.

ISTOP is the number of the last card to be listed.

PRTFIL is the number of the print file.

The first card of the file is counted as zero. If ISTRT does not equal zero, ISTRT cards are skipped at the beginning of the file. Then the cards are listed up through ISTOP. If an end-of-file is deleted during the process, a non-fatal error message is generated giving the card number of where the end-of-file was found.

The routine is called with PRTFIL = 6 for the console printing, and PRTFIL = 20 to yield printed listings.

SUBROUTINE LISTER

This routine controls the list options available to the user at the terminal.

USAGE

CALL LISTER

This routine contains 5 options; CONTENTS, GENEALOGY, COUNT, CARDS, and PRINT. Individual descriptions follow:

| | |
|-----------|---|
| CONTENTS | The routine LISTTC is used to list the table of contents. |
| GENEALOGY | The user is asked to supply the name of the deck to have the genealogy traced. The routine LISTG is called to produce the printout. |
| COUNT | The user is asked to identify the tape where the deck will be found, the tape is rewound, the cards on the file are counted, and the count printed at the terminal. |
| CARDS | The user is asked to identify the tape where the deck will be found and the numbers of the first and last cards to be printed. The subroutine LCRD is used to produce the printout. |
| PRINT | The routine LCRD is used to print, on tape 20 only, the deck on tape 14 with sequence numbers. |

SUBROUTINE LISTG

LISTG prints the genealogy of a specified data deck.

USAGE

CALL LISTG (NAME, IFILE)

where

NAME is the data deck name (2 words, 2A6 format).

IFILE is the FORTRAN logical number of the file on
 which all data resides.

IFILE will be rewound by LISTG.

The genealogy is printed on the standard print file PRTFIL, which
is in common.

SUBROUTINE LISTTC

LISTTC lists the table of contents of the entire data file and prints the list on the standard print file PRTFIL.

USAGE

CALL LISTTC (IFILE)

where

IFILE is the basic data file containing the table of contents
 at the beginning.

If ERFLAG = 0 upon return from LISTTC, there was no error. If ERFLAG \neq 0, an error occurred, such as:

1. End of file detected by subroutine INCD before the \$END OF TABLE card was found.
2. An erroneous card not having one of the formats specified above was found. The offending card is printed along with a message on the standard error file FERR. LISTTC then attempts to complete reading and printing the table of contents.

SUBROUTINE OPT

This routine controls the options printout operation. This operation is provided so that the user will have quick access to a description of the commands available in DORMAN.

USAGE

CALL OPT

This routine lists an option table on the terminal including a numeric alongside of each option. The user then inputs either the word "DONE" to terminate the request, or a numeric to yield a detailed description of the command corresponding to the numeric. The numeric is constrained to be a numeric and bounded by 1 and 12 as it is used in a computed GO TO to branch to the appropriate WRITE/FORMAT statement pair which will describe the command. The process recycles until the user enters "DONE".

SUBROUTINE OUTCD

This routine is used to do all write operations in DORMAN excluding some to the console output file, tape 6, and the write operations done by LISTCD.

USAGE

CALL OUTCD (CARD, IUNIT)

where

CARD is always a 14A6 formatted array.

IUNIT is the tape number that CARD is to be written to.

There are two different output formats handled by OUTCD. They are as follows:

- Mode 1 File is packed tightly with "=" as a 10 column tabbing symbol and ";" as a card terminating symbol.
The output file is blocked in blocks of 4620 characters.
- Mode 2 File is standard FORTRAN output file. (This mode includes a special option of inserting a leading blank for cards written to the console unit 6).

The routine FNDBUF is called to get the mode of IUNIT and pointer to the array IACT. Then the routine branches to process Mode 1 or Mode 2.

Mode 1 processing:

If the flag for end-of-file is set, the routine is terminated with a fatal error message. Otherwise, the card is packed by the routine PACCON and entered into the 4620 character work area. When the area is full, the block is written to IUNIT and the processing continued. When the \$END OF FILE card is encountered, it is also put in the block and the block padded with blanks to fill out the remainder of the 4620 character block. The work area is emptied and an end-of-file written.

Mode 2 processing:

After the end-of-file check, the minimum length of CARD is determined (deleting blanks to reduce I/O time) and the necessary write to IUNIT performed. A check is made for \$END OF FILE card to set an indicator and write an end-of-file.

SUBROUTINE PACCON

This subroutine is used to pack a Hollerith string with the 10 column tabbing symbol "=" and the terminating symbol ";".

USAGE

CALL PACCON (CHAR, OCHAR, NCHAR)

where

CHAR is an input 80 word array of 80A1 format.
OCHAR is the returned array which could be as long as
 81 words in 81A1 format.
NCHAR is the number of packed characters in OCHAR
 array including "=" and ";".

The input string is scanned, all "=" are converted to "-" and all ";" converted to " ". The string is scanned in reverse sequence to obtain the location of the last non-blank character to be used as the first estimate of NCHAR. The array PKTAB is defined as:

PKTAB (I, 1) is the beginning index of a substring. This is
 always 1 greater than a multiple of 10.
PKTAB (I, 2) is the termination index of a substring.

The routine sets up the PKTAB array by scanning each group of 10 characters in CHAR array in reverse order seeking to truncate each substring to a minimum. The beginning and end of the substring are entered into the PKTAB array. The last entry in PKTAB points to the terminating ";".

The shortest PKTAB is for a blank card represented by a single ";".

The groups of characters denoted by PKTAB (I, 1) and PKTAB (I, 2) are moved from the CHAR array to consecutive locations in the OCHAR array accumulating the variable NCHAR with each move.

SUBROUTINE REPL

This routine controls the operation that replaces decks in the data base file.

USAGE

CALL REPL

This routine communicates with the terminal to obtain the names of the two decks to be used in replacing deck two with deck one in the data base file. The two deck names must be identical. The name for deck one must be either the name of a mod deck on tape 13 or a basic deck on tape 14. The request for replacement is rejected if neither condition is met. The subroutine REPLAC is used to do the replacement operation.

SUBROUTINE REPLAC

This routine replaces a deck on the basic file IN1 by another deck of the same name which is found on file IN2.

USAGE

CALL REPLAC (IN1, IN2, OUT)

where

IN1 is the file on which all existing data resides.

IN2 is the file containing the substitute deck.

OUT is the file on which the modified data will be written.

Files IN1 and OUT are rewound by REPLAC upon entry. File IN2 is not rewound but rather is assumed to be correctly positioned at the beginning of the deck to be substituted.

The first card of the new deck must be

\$DECK NAME1

or

\$DECK NAME1 USES NAME2

The routine finds the existing deck NAME1 which resides on file IN1 and replaces it with the version which is on file IN2. The \$DECK card, both in the new deck and in the table of contents, contains the date and time of the replacement, in Hollerith. If the new deck uses another deck (which may differ from the deck used by the original deck), that deck must already exist on the data file. The new deck must end with the card \$END OF DECK.

The error flag ERFLAG is set to zero upon return from REPLAC, if no errors occurred, to 1 if an error occurred. Error messages are written on file FERR. Possible errors are:

1. First card of new deck is not \$DECK.
2. Deck to be replaced cannot be found on file IN1.
3. New deck uses a deck which does not exist on file IN1.

The deck to be inserted is considered to terminate with the first \$END OF DECK card encountered on file IN2. In the event of an error, REPLAC advances file IN2 to the end of the new deck (unless a premature read error occurs on file IN2).

SUBROUTINE RESTOR

This subroutine repositions a card file at a designated spot. The first card to be in the buffer is specified and the buffer is full upon return. The routine FILBUF is used to obtain card images.

USAGE

CALL RESTOR (BUF, NC, NL, FILE, NCNOW)

where

| | |
|-------|--|
| BUF | is the buffer to be filled. |
| NC | is the number of the card in the file to be put at position 1 in BUF. |
| NL | is the number of cards in the buffer. NL is the capacity of the buffer (less if the \$END OF DECK has been encountered). |
| FILE | is the tape number to be read for the card images. |
| NCNOW | is the number of the card currently at position 1 in the buffer. |

The routine has three modes of operation:

1. If $NC = NCNOW$, FILE is already positioned, and the subroutine returns.
2. If $NC > NCNOW$, FILE is read forward until $NC = NCNOW$; the buffer is filled.
3. If $NC < NCNOW$, FILE is rewound; step 2 is performed to advance the file and fill the buffer.

SUBROUTINE SAVER

This routine controls the save operation.

USAGE

CALL SAVER

This routine communicates with a terminal to determine which of the three save options is to be performed; DATA BASE, DORCA or MOD DECK. Failure to input a recognizable request terminates the request. The three operations are separate and distinct.

1. When a data base is to be saved, the program requires that the data base be either created or modified by addition, deletion or replacement. The user is requested to supply the version identifier to be put in the first card of the deck. The data base is then labeled with ID, date and time, and copied to tape 4.
2. When a DORCA data deck is requested, the program asks the user to provide the name of the deck to be saved. If the deck is not the same as the deck on tape 14, then the user is requested to provide destruct permission to assure that no error was made in the name. If permission is a "NO", then the request is terminated. Otherwise, the deck is extracted from the data base with the routine USE. The deck then has the first and last cards removed as the deck is copied onto tape 12.
3. If a mod deck is to be saved, the routine asks for the name of the mod deck. If it does not match the name on the mod deck on tape 13, the request is terminated. Otherwise the mod deck is copied onto tape 24.

SUBROUTINE SYNCBF

This subroutine seeks a match between two card files by using a multi-buffer search.

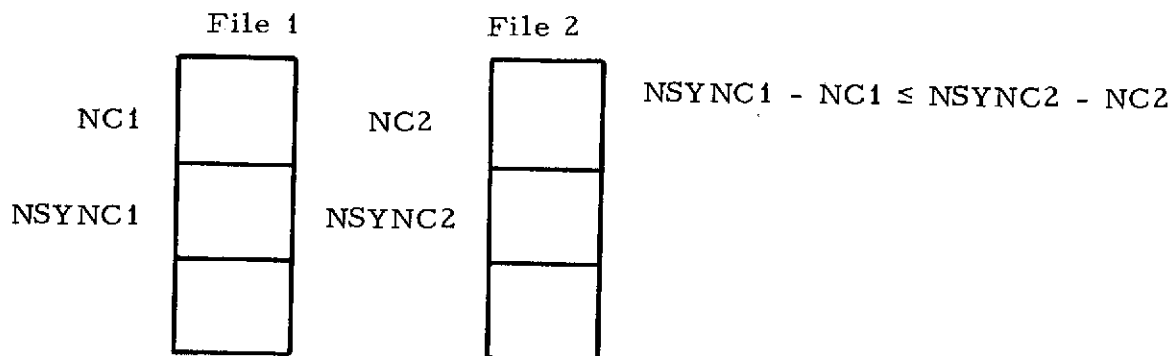
USAGE

```
CALL SYNCBF (BUF1, NC1, NL1, FILE1, NSYNC1, ISYN1,  
             BUF2, NC2, NL2, FILE2, NSYNC2)
```

where

| | |
|----------------|---|
| BUF1, BUF2 | are the two buffers available. Both will be filled on entry. |
| NC1, NC2 | are each the number of the first card in the buffers (known mismatch). |
| NL1, NL2 | are the number of cards in the buffers. |
| FILE1, FILE2 | are the tape numbers where the cards are located. |
| NSYNC1, NSYNC2 | are the card numbers in the file where the cards match (returned parameters). |
| ISYN1 | pointer to the sync array ISYN. |

A match can be found only for the following diagram:



Upon entry, the two buffers are full and known to have no match by the rules of SYNC1. The buffer for file 1 is filled with the next batch of cards. Now a match can be attempted for every card in BUF1 with every card in BUF2. If no match is found, BUF1 is filled with the next batch of cards from file 1. The search is now constrained by the rules of SYNC1 ($I = 1, N; J = 1, I$). If no match, BUF1 is reloaded with the original set of cards (beginning at NC1), and BUF2 is filled with the next batch of cards from file 2. The cycle then repeats.

The process can be long and time consuming, but will always terminate on the \$END OF DECK card at the very end of both files.

SUBROUTINE SYNCDS

This routine processes the input of the sync cards used in the deck differencing process.

USAGE

CALL SYNCDS (FSYN)

where

FSYN is the tape number to be read for the card input
 (set elsewhere to be the console unit 5).

The user is requested to input the sync cards. The file FSYN is read in alphabetic mode via INCD to obtain each sync card, until either the limit of 20 is reached or the word "DONE" is encountered. Each pair of numerics is translated via the routine VALUE to assure correctness of entry. The card is reinput if the data is not numeric or "DONE".

SUBROUTINE SYNC1

This routine attempts to find a match in the buffers when a mismatch has been found by DIFDEC.

USAGE

CALL SYNC1

where all necessary variables have been communicated via labeled common.

This routine is coded on the assumption that most decks are similar and that matching card pairs are in close proximity to one another. The buffers are moved so that the original mismatched pair of cards are at position 1 in the buffer and the buffers refilled. The process is best described by an example. If a mismatch occurred at pair LIN1, LIN2, and, if buffer 1 has not been matched after 4 attempts, then on attempting to match the 5th card in BUF1 (including LIN1), the cards in BUF2 available for comparison are numbers 1, 2, 3, 4 and 5 (including LIN2). Eventually, a match does occur and the positions of the match are saved. The process is repeated with the two buffers effectively interchanged. The positions of the match are again saved. At this point the routine has two potential mods it can select. Selection is based on the minimum total change, insertions, plus deletions, for the two mods. The necessary insertion/deletion cards are issued to the file FOUT.

If no match is found, a call is made to SYNCBF to locate a match in a multi-buffer search mode.

SUBROUTINE TERM

This routine terminates the run in response to the user providing the word "DONE" in response to ENTER OPTION REQUEST, or in the event a fatal error has been detected by the program.

USAGE

CALL TERM

This routine terminates the computer run with the message:

- - END DORMAN - - RUN TERMINATED

If the error flag ERFLAG is non-zero when the routine is called, the message ERROR IS FATAL is also printed.

SUBROUTINE UNPAC

This routine unpacks cards containing "=" and ";".

USAGE

CALL UNPAC (A, B, N)

where

A is the 84 word input character string of format 84A1.

B is the 84 word output character string of format 84A1.

N is the number of characters found in string A.

The array B is set to blanks. For each character in A up to the ";", the counter N is set and the character moved from A to B. If the character in A is ";", the process is terminated. If the character in A is "=", the location of the next character in B is adjusted upwards to be 1 greater than the next multiple of 10. The loop then cycles to move the next character.

If the card image exceeds 84 characters, a fatal error message is generated and DORMAN is terminated via a call to TERM.

SUBROUTINE UREAD

This routine is used to read all 84 column cards on the CDC series equipment. On the UNIVAC 1108 the routine is used only for single card images, and the reading of the 4620 character buffers is taken care of in the INCD routine.

USAGE

CALL UREAD (IUNIT, ICRD)

where

IUNIT is the tape number to be read for a card.

ICRD is a 14 word array in 14A6 format.

A card is read from unit IUNIT into the array ICRD. The return is done, unless an end-of-file is encountered. In that case, the \$END OF FILE card is generated and control is then returned.

SUBROUTINE USE

Subroutine USE extracts a deck from the data base and puts it on the final data file FINAL, which is in common.

USAGE

CALL USE (NAME)

where

NAME is the data deck name (2 words, 2A6 format).

All data is assumed to reside on the basic data file BASIC, which is in common. USE obtains the genealogy of deck NAME from the table of contents at the beginning of the BASIC file. This genealogy consists of one basic deck and possibly a chain of one or more mod decks:

NAME MOD1 MOD2 . . . MODN BASICX

USE copies all mod decks, if any, from BASIC onto tape 27, and positions BASIC at the beginning of the basic deck needed for NAME. Then EDITDK is called to combine the decks and place the result on FINAL on 22 and 23 (if multi-pass mod decks).

I/O files used:

| | |
|---------------|---|
| BASIC | basic input data file (rewound by USE). |
| FINAL | final output data file (rewound by USE). |
| MTAPE, S1, S2 | scratch tapes. Logical numbers are defined in data statements within USE. |
| FERR | file for error messages. |

Errors:

If no error, ERFLAG = 0 upon return from USE. If ERFLAG = 1, USE could not locate on BASIC one of the mod or basic decks required by NAME, or there was an error discovered by the EDITDK subroutine. In case of error, a message is printed on the standard error file FERR.

SUBROUTINE USER

This routine controls the extraction of decks from the data base file, or the application of a new mod deck.

USAGE

CALL USER (IFLAG)

where

IFLAG is zero for first call only and is externally set.

If IFLAG is not zero, the message CREATE OPTION IS BLOCKED is printed.

This routine communicates with the terminal to obtain the name of the deck to be extracted and placed on tape 14. The name is first checked to see if it corresponds to a new mod deck on tape 13. If so, the basic deck obtained by the application of the mod deck via the routine EDITDK is put on tape 14. Otherwise, the name deck is extracted from the data base file by the routine USE. Tape 23 is used as a scratch file, if tape 14 contains the basic deck required by the mod deck on tape 13.

SUBROUTINE VALUE

VALUE converts numeric data from Hollerith to floating point format.

The coded value is assumed to be stored in two successive cells in (A6, A4) format, representing 1 to 10 digits with an optional decimal point. The value may be located anywhere in the 10-character field, but must not contain embedded blanks. No exponents or plus or minus signs are permitted. The calling sequence is:

CALL VALUE (A, V, IERR)

where

| | |
|------|--|
| A | is a 2-cell array containing the coded value in (A6, A4) format. |
| V | contains the converted value in floating pointing, upon return. |
| IERR | is an error flag, upon return. IERR = 0 for no error. IERR = -1 if the field was completely blank. (A blank field is <u>not</u> taken to represent zero.) IERR = 1, 2, ..., 10 to indicate position of an illegal character, multiple decimal points or embedded blank. |

The purpose of this routine is to detect input errors in numerical entries without causing a machine abort. Thus, this routine examines the input value while it is still in coded format to determine if there are any illegal aspects that would cause an abort.

The unpacking and repacking of the coded word into 10 separate characters and the ultimate conversion to floating point are accomplished by calling the system routines ENCODE and DECODE, which operate similarly on both the CDC 6000 and UNIVAC 1108 machines, though with slightly different calling sequences. The array C, in which the 10 separate characters are stored, consists of 19 cells to facilitate right-adjusting the non-blank characters, as is necessary before converting to floating point.

FORMAT OF BASIC DATA FILE

The following is a description of the format of the basic data file. It is provided as a means of reference for the process of validating the correctness of a data file for DORMAN.

The file, documents, and the program deck are all that need be sent from one individual to another. The data file is composed of the following elements:

1. Version identifier card
2. Table of contents
3. Mod decks
4. Basic decks
5. End-of-file card
6. Single end-of-file.

The version identifier card is a single card at the beginning of the file. It is structured as

```
$DORMAN DATA BANK  VERSION XXXXXX  TIME  DATE
```

where

XXXXXX is a 6 character identifier.

TIME is the time of day the file is created.

DATE is the date the file is created.

The table of contents is a set of deck cards of the form

```
$DECK  NAME1  TIME  DATE
```

or

```
$DECK  NAME2  USES  NAME1  TIME  DATE
```

where

NAME1 is the name of a basic deck.

NAME2 is the name of a mod deck.

| | |
|------|--|
| TIME | correspond to when the deck was entered into |
| and | |
| DATE | the file. |

When adding a deck to the bank, the table of contents is also updated, subject to two rules:

1. If the deck is a mod deck (contains the word USES), the deck card is entered as the first element of the new table of contents.
2. If the deck is a basic deck, the deck card is entered as the last element of the new table of contents.

Mod decks begin with a \$DECK card containing the word USES, consist of \$DELETE, \$INSERT, \$CHANGE and DORCA data deck cards, and terminate with a \$END OF DECK card. The function of a mod deck is to reduce the size of the data file by using as a reference another DORCA data deck that is very similar in content. The basic deck NAME1 must be in the data file. Mod decks can be chained as a sequence of mod decks. When a mod deck is added to the data file, it is placed as the last of the set of mod decks.

Basic decks are DORCA data decks varying from 1200 to 2200 cards in size. They begin with a \$DECK card and terminate with a \$END OF DECK card. When a basic deck is added to the data file, it is assumed to be more likely to be used than an old basic deck and, therefore, the new basic deck is the first basic deck in the set of basic decks.

The end-of-file card is \$END OF FILE.

FILE DEFINITIONS

The following is a definition list of what tapes are used for in the various procedures of DORMAN.

| | |
|---------|---|
| TAPE 1 | Input source of previously existing data file. |
| TAPE 2 | Tape 1 is copied to Tape 2, if any decks added or deleted. Tape 2 is initial creation data file. |
| TAPE 3 | Tape 2 is copied to Tape 3, if any further decks added or deleted. |
| TAPE 4 | Tape reserved for saving data file. |
| TAPE 5 | Console input unit. |
| TAPE 6 | Console output unit. |
| TAPE 11 | Input mod decks from external source. |
| TAPE 12 | Tape reserved for output of DORCA data deck. |
| TAPE 13 | Input resultant mod deck from deck differencing process. |
| TAPE 14 | Current basic deck resides on this unit. |
| TAPE 20 | Print file for listings of decks. |
| TAPE 21 | Input basic decks. |
| TAPE 22 | Scratch file for EDITER, CONV. |
| TAPE 23 | Scratch file for USER, EDITER, CONV. |
| TAPE 24 | Output mod decks. |
| TAPE 25 | Scratch tape for USE. |
| TAPE 26 | Scratch tape for USE. |
| TAPE 27 | Scratch tape for USE. |

LIST OF COMMONS WITH VARIABLES

All variables used in this program are integers except for one parameter used in calls to the subroutine VALUE and one logical variable.

The following is a list of label common blocks with a description of each variable used. Most of the variables are initialized by data statements in the block data subprogram.

COMMON/MISC/

| | |
|--------|--|
| ERFLAG | Error indicator set not zero if an error has occurred. |
| FERR | Error printout file (equivalent to PRTFIL), set to TAPE 6. |
| KARD | Array for card images. |
| ACTION | Alternate array for card images. |

COMMON/NAMES/

| | |
|--------|--|
| IVER | Version of data base file, set to 4HTEMP. |
| DNAME | Array with name of saved DORCA data deck on tape 12. |
| MODNAM | Array with name of mod deck on tape 13. |
| BNAME | Array with name of basic deck on tape 14. |

COMMON/FILES/

| | |
|-------|---|
| BASIC | Number of tape where current data base file resides (1, 2, or 3). |
| MTAPE | Scratch tape variable. |
| FINAL | Number of tape where basic deck is located. |
| S1 | Scratch tape variable. |
| S2 | Scratch tape variable. |

COMMON/REST/

| | |
|-------|-------------------|
| TABLE | Set to 6HF TABLE. |
| USES | Set to 6H USES. |
| FILE | Set to 6HF FILE. |
| END | Set to 6H\$END O. |
| DECK | Set to 6HF DECK. |
| DDECK | Set to 6H\$DECK. |
| BLANK | Set to 6H. |
| BATCH | Set to zero. |

COMMON/VCARD/

| | |
|------|--|
| ILBL | Array set up for label card of data base file. |
|------|--|

COMMON/WORK/

| | |
|--------|--|
| IIUNT | Array containing a maximum of 3 integers for 3 concurrently operating tapes. |
| IIRW | Array of corresponding read write statuses. |
| IUTBL | Array of files with status and mode indicators available for program. |
| IACT | Array to save pointers for files between calls to OUTCD and INCD. |
| NFILES | Maximum number of files available in DORMAN, set to 19. |

COMMON/BUFFER/

| | |
|-----|---|
| MAX | Maximum number of cards that can be in internal buffer at a given time. |
|-----|---|

COMMON/BFRS/

This common has two definitions:

| | |
|-------|--|
| ISYN | Array of sync points in two decks. |
| BUF1 | Card buffer for file 1. |
| BUF2 | Card buffer for file 2. |
| CN1 | Number of current card being processed in BUF1. |
| CN2 | Number of current card being processed in BUF2. |
| NB1 | Number of cards in BUF1. |
| NB2 | Number of cards in BUF2. |
| NWBUF | Maximum number of card images in either buffer (agrees with dimension of BUF1 and BUF2). |
| LIN1 | Number of card in file 1 at first position in BUF1. |
| LIN2 | Number of card in file 2 at first position in BUF2. |
| ICN1 | Scratch variable. |
| ICN2 | Scratch variable. |
| ISYN1 | Index to current pair of sync numbers. |
| IT1 | Scratch variable. |
| IT2 | Scratch variable. |
| IT3 | Scratch variable. |
| FULL | Logical variable set TRUE if both buffers are full. |

The other definition is:

| | |
|-------|---|
| IMOD1 | Array for 2 working areas for reading and writing of mod 1 files. |
| IWK1 | Working area for mod 1 card images. |

| | |
|--------|-------------------|
| ITEMP1 | Scratch array. |
| ITEMP2 | Scratch array. |
| IIX | Scratch variable. |
| IY | Scratch variable. |
| IIZ | Scratch variable. |

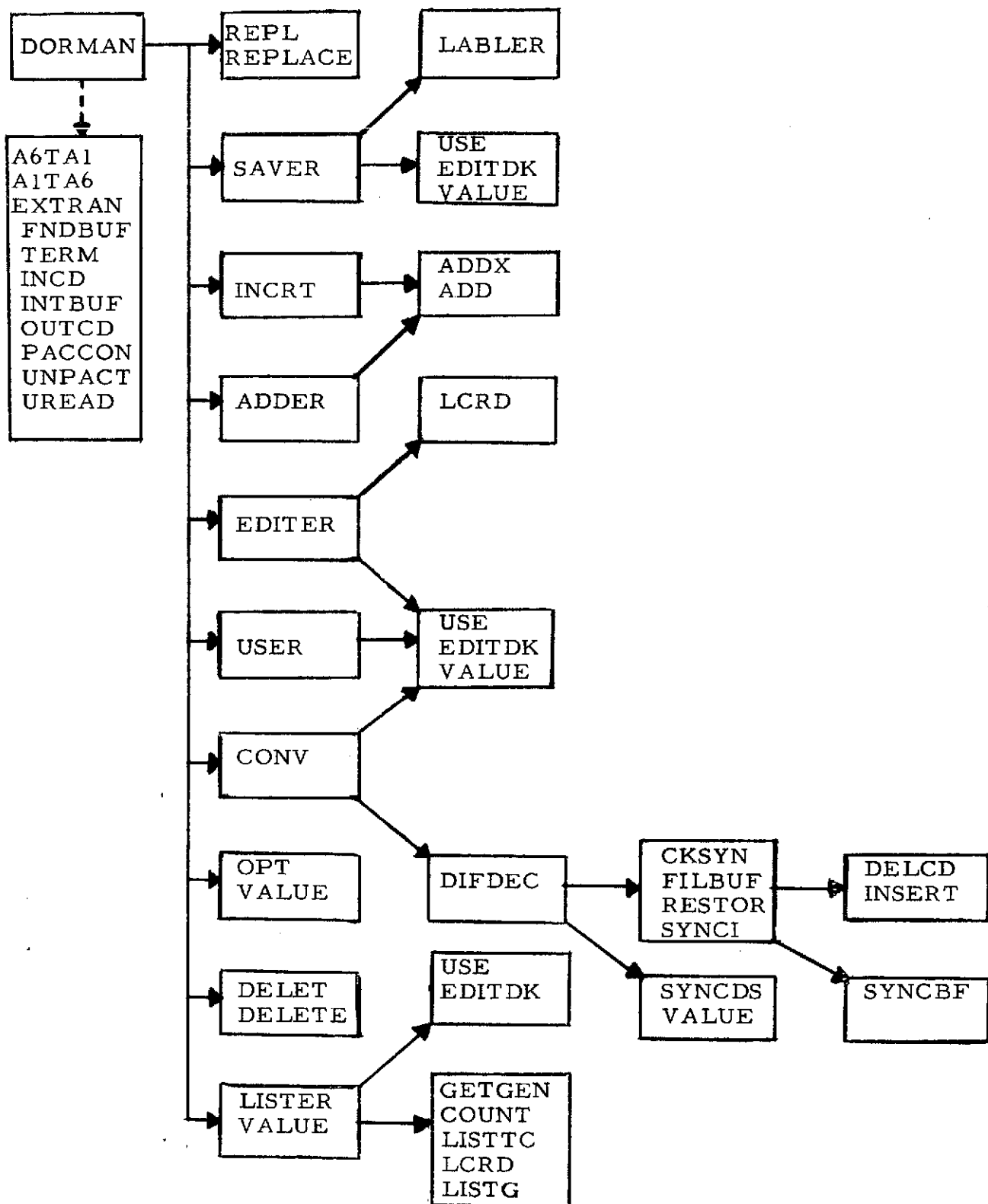


Figure 1. Segmentation/Linkage Map